Q2]

A) An algorithm with complexity knlog(kn)

Using merge sort:

- We implement the combining phase of merge sort for k different arrays where total number of elements is k*n
- Therefore, on each iteration, we will have half the number of arrays we had in the previous iteration.
- Imagine a tree where the k arrays merge together, 2 at a time using merge sort.
- The length of the tree will be **log(nk)**
- Also, we know that the merging phase of merge sort takes **O(n).** Here, we have a k*n elements. Therefore, merge sort will take **O(kn)** for merging in total
- Therefore, the total time complexity: **O(k*n*log(nk))**

B) Second algorithm for kn(logk)

Using Min-Heap:

- Select the first elements from all the k arrays and add them to a min heap.
- Since the k arrays are sorted, the first elements selected from all the arrays are the lowest elements in their respective array.
- Thus, the root of the element in the min-heap is the smallest element among all the k arrays. Extract the root and save it in the output array of size k*n (total number of elements) and heapify the min heap.
- Now, add the next element in the min-heap from the array where the root element extracted from the min heap in the last iteration belonged to.
- Thus, keep extracting the root from the min-heap as you keep adding the elements one by one from the k arrays
- Eventually, we will have a sorted output array of kn elements.
- Insertion / Deletion operation in minheap requires **O(logk)** time. Since we have kn elements, the total time required will be **O(knlog(k))**